

|  |
| --- |
| **DeepPi** |
| Deteção incêndios através de Deep Neural Learning |
| Relatório de Projeto Aplicado submetido como requisito parcial para obtenção do grau de Mestre em Computação Móvel |
| Orientador: Professor Carlos Carreto  Coorientador: Professor Adérito Acaso |
| **Diogo Manuel de Ascensão Almeida Aleixo** |
|  |
| **03 | 2016** |

If i have seen further than others, it is by standing upon the shoulders of giants

[Isaac Newton]

**Agradecimentos**

O trabalho apresentado no presente relatório só foi possível graças à colaboração e apoio de algumas pessoas, às quais não posso deixar de prestar o meu reconhecimento.

À minha namorada por sempre me ter apoiado incondicionalmente em todas as decisões da minha vida, tendo sido nesta fase um elemento essencial para o término do trabalho.

Agradeço também ao Adrian Rosebrock pelo excelente curso sobre visão computacional. Este curso foi sem duvida a maior mais valia para a minha aprendizagem no tema e serviu de apoio para construir o software. O curso é destinado a cientistas de dados projetado para 10 meses. Passa por temas como processamento de imagem do básico ao avançado, aprendizagem máquina, análise de dados em *Big Data* entre outros temas interessantes.

Aos autores do curso *cs231n* disponibilizado por *Stanford* e ao Michael Nielsen pelo excelente livro *Open-Soure* disponibilizado.

À minha mãe e irmãos por me terem dado a força e os ensinamentos necessários durante toda a vida para cumprir com sucesso esta etapa.

Ao meu filho que apesar da tenra idade foi um elemento essencial onde arranjei a motivação necessária para acabar o projeto para obtenção do grau de Mestre em computação móvel.

Por fim, e não menos importante, a todos os meus amigos que me acompanharam neste processo e o influenciaram direta ou indiretamente.

**Resumo**

O fogo é um desastre ambiental e/ou humano que pode atacar em qualquer lugar e ser bastante destrutivo, consumindo bens materiais, ambientais e humanos. Métodos existem para detetar fumo e fogo e permitir as autoridades atuar de modo a evitar ou minimizar os danos. Os métodos existentes vão desde a utilização de sensores do meio ambiente, à utilização de camaras, sejam elas de *CCTV(Closed circuit television)* ou projetadas para a deteção do fogo. O objetivo deste trabalho foi, a aplicação de algoritmos de processamento de imagem e *machine learning* para a deteção de fogo bem como o desenvolvimento de um módulo de energia renovável através de painéis fotovoltaicos

Foi também objetivo o ajustamento do objetivo principal a Hardware de baixo custo de modo a reduzir o preço final do sistema e a recolha das imagens através de um UAV.

**Abstract**

Fire is an environmental and / or human disaster that can strike anywhere and be very destructive , material, environmental and human assets consuming. Methods exist to detect smoke and fire and allow the authorities to act in order to avoid or minimize damage. Existing methods range from the use of environmental sensors, the use of cameras , whether or CCTV *(Closed circuit television)* designed for the detection of fire. The objectives of this work are the application of image processing algorithms and machine learning algorithms for fire detection and the development of a renewable energy module

It is also aimed at the adjustment of the main objective of low cost hardware to reduce the final price of the system.

# Siglas(EM progresso – Falta descrição)

|  |  |
| --- | --- |
| Android |  |
| Background |  |
| Backpropagation |  |
| Bias |  |
| CCTV |  |
| Centroid |  |
| CO2 |  |
| Feed forward |  |
| FeedForward |  |
| Foregroud |  |
| FTS |  |
| Gps |  |
| Gradient descent |  |
| Hidden Layers |  |
| Ios |  |
| LIDAR |  |
| Machine Learning |  |
| MMS |  |
| Numpy |  |
| OpenCv |  |
| Pan&Tilt |  |
| Python |  |
| Raspberry Pi |  |
| Recall |  |
| RGB / HSV / L\*a\*b |  |
| Scikit-learn |  |
| SkLearn |  |
| SMS |  |
| Streamming |  |
| SVM |  |
| Threshold |  |
| UAV |  |
| Valores Raw |  |

# INTRODUÇÃO

Este capitulo tem como objetivo fazer uma introdução ao projeto e contextualizar o mesmo no âmbito da engenharia informática e a sua motivação. Também neste capitulo é descrito o problema a resolver, a solução proposta para a sua resolução e contribuição para projetos semelhantes. Por ultimo é exposta a organização dos capítulos que constituem o relatório

## Enquadramento e motivação do projeto

Neste relatorio é descrito um sistema para deteção de fogos. Este sistema é enquadrado na área técnica do processamento de imagem, inteligência artificial, aprendizagem máquina e energias renováveis.. A principal motivação deste projeto é, como será descrito no próximo ponto do atual capitulo, a crescente destruição de recursos naturais e humanos pelos fogos em todo o mundo. Este facto torna urgente uma solução capaz de solucionar de alguma forma este problema. Um fogo pode causar mortes por inalação, queimaduras de vários graus (levando no caso mais extremo à morte), consumir vários hectares de terreno numa questão de horas, consumir bens materiais, entre outas cosias.

Nesse sentido, este projeto tem como objetivo ajudar os bravos bombeiros e pessoas a quem afeta os fogos afetam direta ou indiretamente.

## Definição do problema e objetivos

Em Portugal e em todo o mundo, os incêndios consomem todos os anos vários milhares de hectares. Motivados pela ganancia ou por um simples descuido, estes tendem a consumir áreas verdes e casas de população da zona tornando-se um grande perigo para a população em geral. Os bravos bombeiros, consomem muitos recursos e por vezes infelizmente vidas no combate deste desastre. Hoje em dia os incêndios são extinguidos com recurso a meios terrestres e aéreos que saem também eles muito caros aos cofres do estado de qualquer nação. A deteção do incendio é atualmente muito primitiva, limitando-se a equipas estrategicamente posicionadas na floresta, a sistemas de sensores distribuídos e a visão computacional. O método de equipas na floresta, sendo humano é falível. O humano, por natureza, distrai-se com facilidade quando deparado com situações monótonas. Os sensores distribuídos pela floresta têm o problema de se limitarem a fazer a função de sensor, e “dispararem” somente quando existe e é detetado um valor anormal de *CO2* (Dióxido de carbono) por exemplo, o que pode ser tarde alem de o resultado ser influenciado pela direção do vento. Os sensores têm também o problema de o diâmetro de atuação ser limitado, obrigando a um investimento muito grande para uma área considerável. As camaras com poder de visão computacional têm o problema do custo e do raio de visão, que apesar de poder ser grande, pode ser influenciado com uma arvore grande, ou um rochedo. Para evitar tais problemas, estas são muitas vezes colocadas em cumes e a analisar a encosta adjacente. O problema desta solução é que embora resolva o problema em grande parte, continuam a haver muitos ângulos mortos.

Após descritos os problemas existentes pretende-se que o utilizador usufrua das seguintes funcionalidades do sistema através de um sistema low-cost:

* Deteção de fogo;
* *Streaming* do vídeo em tempo real da camara;
* Camara com visão 360 graus;
* Alertas através de notificações remotas;
* Sistema energeticamente autossuficiente;

O output final será a validação do funcionamento de uma rede neuronal no Raspberry Pi para o objetivo descrito com as funcionalidades descritas.

## Solução proposta para resolver o problema

A solução proposta para a resolução do problema mencionado anteriormente foi a captação de imagens através de uma camara de baixo custo com movimento de 360 graus, imagens essas que serão posteriormente processadas por uma rede neuronal convolucional ou CNN. O sistema é alimentado por um módulo de energia renovável solar.

Foi montada uma camara ligada a um *Raspberry* *pi* 3 com capacidades de *pan&tilt* que por sua vez está dotado de um portal web que permite a utilizadores com permissões aceder ao streamming das imagens captadas pelo sistema

De notar que o *streaming* só é feito quando o utilizador pede, de modo a evitar a transmissão excessiva de dados. No entanto, tudo o resto, continua a trabalhar em background.

Foi também desenvolvida uma aplicação para *Android* que permite a receção de notificações remotas através do serviço *Firebase* da *Google* e a visualização do site web. De modo a tornar o sistema autossustentável, foi criado um módulo que permite atribuir uma certa autossuficiência ao sistema através de painéis fotovoltaicos.

O objetivo deste projeto foi criar um sistema que fosse capaz de detetar fogo e fumo através de vídeo em tempo real, conseguir com sucesso avisar as autoridades competentes ou dono do sistema com o mínimo de atraso possível e conseguir enquadrar o sistema em hardware de baixo custo. O sistema foi feito para detetar fogos na fase o mais inicial possível. O fogo é um dos poucos desastres em que os danos podem ser prevenidos ou minimizados quando comparando com outros desastres naturais como furacões, tsunamis ou terramotos. Para a prevenção ou minimização dos danos, o fogo tem de ser extinto na fase mais inicial possível da sua vida, de modo a evitar que este cresça e assim se torne difícil extinguir.

O sistema deve ter uma resposta o mais rápido possível de modo a permitir as autoridades competentes atuar precocemente sobre o incidente. Não faz sentido avisar sobre um incendio depois de ter consumido uma larga área.

## Contribuição do projeto para o estado da arte

A deteção de fogo e fumo é um tema muito vasto, tendo pela comunidade cientifica já vários estudos publicados e disponíveis sobre diferentes técnicas.

Foram estudados métodos de deteção através de sistemas *LIDAR(Light detection and ranging)* sensores, processamento de imagem e algoritmos de *machine learning*.

A contribuição para o estado da arte da temática abordada vai no sentido da exploração de diferentes combinações para processamento de imagem e redes neuronais convulsionais Também, o projeto, tem como objetivo, ser implementado em hardware low-cost, sendo constituído por um *Raspberry* *pi* 3 capacitado de uma camara, e um motores para a função de pan&tilt, alimentados por uma bateria que por sua vez é também ela alimentada por painéis solares

Este sistema, contribuí para o estado da arte com a implementação de uma rede neuronal convolcional para análise de imagem num dispositivo de baixo custo e fraco processamento.

## Organização do relatório

**O presente relatório encontra-se dividido em 7 capítulos.** No primeiro capitulo é feita uma introdução ao trabalho desenvolvido, definido o problema, e proposta uma solução para a sua resolução.

O segundo capitulo, é destinado ao estudo da técnica do projeto, mencionando com detalhe os projetos estudados ao longo do trabalho que envolvam tecnologias e/ou objetivos semelhantes. É também no segundo capitulo, feita uma comparação entre diferentes técnicas.

O terceiro capitulo é dedicado à exposição do estudo efetuado durante a realização do trabalho exposto no presente relatório. Neste capitulo é são explicados os dois temas de maior relevância neste trabalho. O processamento de imagem, técnica que permite extrair indicadores de uma imagem, e a aprendizagem máquina que permite atribuir a um programa a habilidade de aprender.

O quarto capitulo apresenta a análise de requisitos feita do sistema. Neste capitulo são detalhados os requisitos técnicos ao funcionamento do sistema.

No quinto capitulo é exposto ao pormenor o sistema desenvolvido. Aqui, é apresentada a arquitetura da solução final e descrição dos módulos do sistema.

O sexto capitulo é destinado a testes e resultados. Depois de serem cumpridos os objetivos do trabalho, o mesmo foi exposto a um conjunto de testes de modo a comparar resultados com resultados já existentes e poder dar uma contribuição para a comunidade cientifica, conseguido desta forma afirmar o método aqui exposto na realização do trabalho.

No sétimo e ultimo capitulo é feito um resumo de conclusão do trabalho assim como apresentados futuros melhoramentos do sistema.

# Trabalhos relacionados

No presente capitulo é descrito o objetivo da visão computacional e trabalhos relacionados com objetivos semelhantes. Foram estudados sistemas desenvolvidos com fins académicos ou como prova de conceito (Cap. 2.1) e sistemas comerciais existentes e com resultados comprovados (Cap. 2.2).

## Exemplos de trabalhos semelhantes

A metodologia para escolher o conjunto de sistemas e algoritmos utilizados no projeto, foi um estudo de técnica.

Foram estudados sistemas comerciais existentes e estudos sobre o tema. Os artigos que foram estudados dividem-se em várias categorias que serão explicadas no próximo subcapítulo:

* Métodos baseados em *Bayesian* networks;
* Métodos baseados em máquinas vetoriais;
* Métodos baseados em redes neuronais;
* Métodos baseados em sistemas *LIDAR*;
* Métodos baseados em sensores;

Após o estudo efetuado, ficou percetível de que existem vários meios para atingir o mesmo fim. Da mesma forma, todos têm as suas particularidades e desvantagens.

### Métodos baseados em Bayesian Networks

O sistema proposto [Bayesian Networks] utiliza redes probabilísticas ou Bayesian Networks. A verificação é feita em vários passos como demonstra a imagem seguinte.

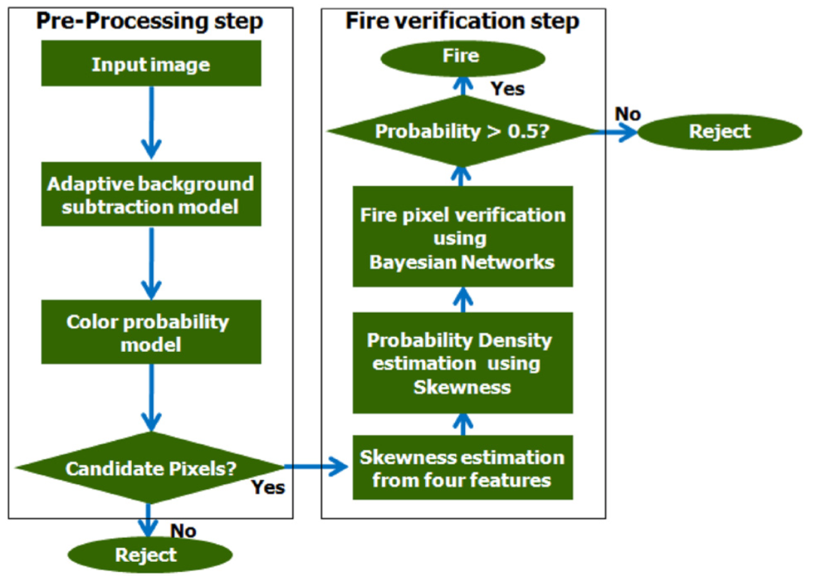


Figura 1-Bayesian Networks, algoritmo

Em primeiro lugar é separado o background do *foreground*. Com esta técnica o autor do artigo consegue separar os pixéis candidatos a pixéis de fogo, dos pixéis que não são de fogo. Esta técnica é possível porque é sabido que o fogo está em constante alteração de forma, e assim é possível construir um algoritmo para identificar tudo o que altera a forma como sendo de *foreground*

A segunda técnica, prende-se com a deteção de pixéis com as mesmas cores da gama do fogo. Mesmo após a separação do background e do *foreground*, e por não ser linear que seja só o fogo detetado como *foreground*, são também separados pixéis que não pertencem a essa categoria. Para conseguir filtrar os pixéis considerados barulhos na imagem, o autor utiliza um mapa de cores em probabilidades uni modais de Gauss. Essas probabilidades são calculadas através de um *dataset* de imagens que contêm fogo. O pixel é depois comparado com a probabilidade e o algoritmo capaz de decidir se este é ou não fogo.

Após a ultima verificação, o autor utiliza um módulo de redes probabilísticas. Apesar da filtragem baseada em processamento de imagem, o algoritmo, segundo o autor do artigo, não era capaz de distinguir objetos da cor vermelha em movimento, de fogo. Foram feitos testes com vários filmes em ambientes diferentes como mostra a imagem seguinte.

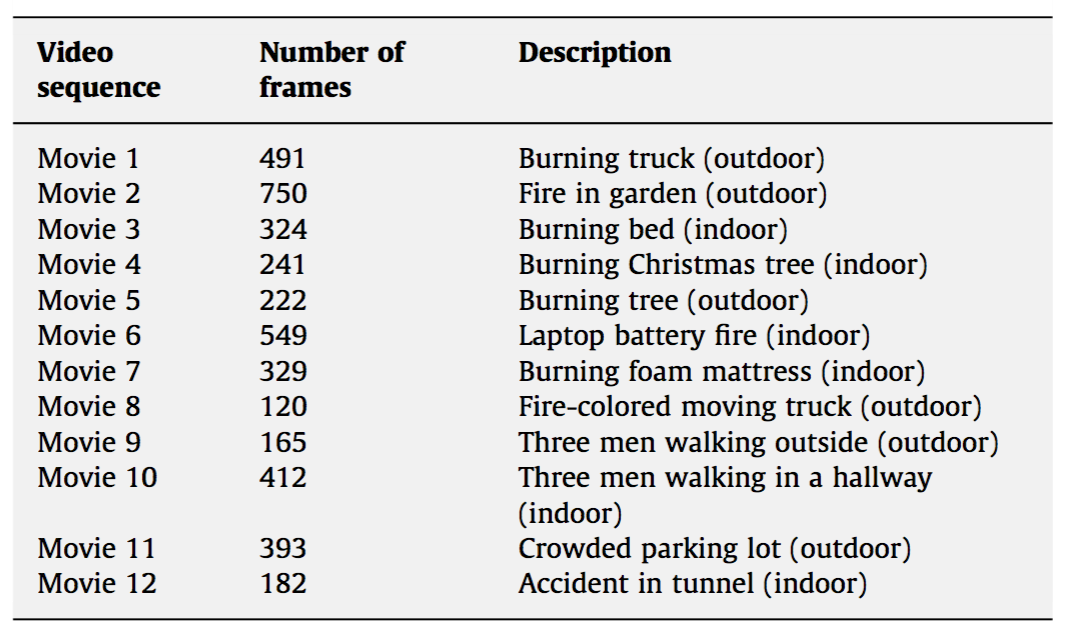


Figura 2-Bayesian Networks, dataset

E comparados os verdadeiros positivos e falsos positivos com dois métodos descritos no artigo (*Toeyin* e de *Ko*) como mostram as duas imagens seguintes.

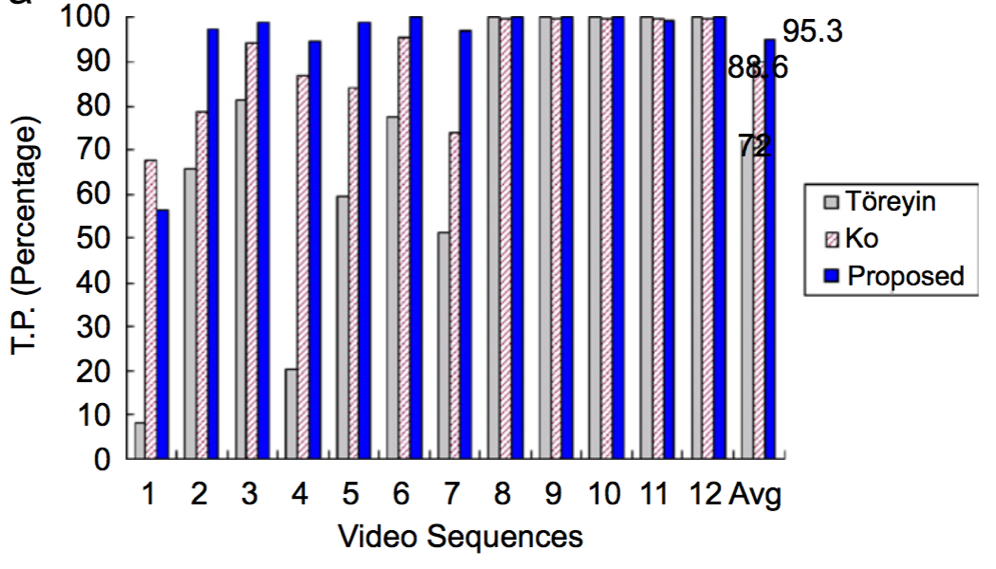


Figura 3-Bayesian Networks, comparação com outros métodos (Verdadeiros positivos)

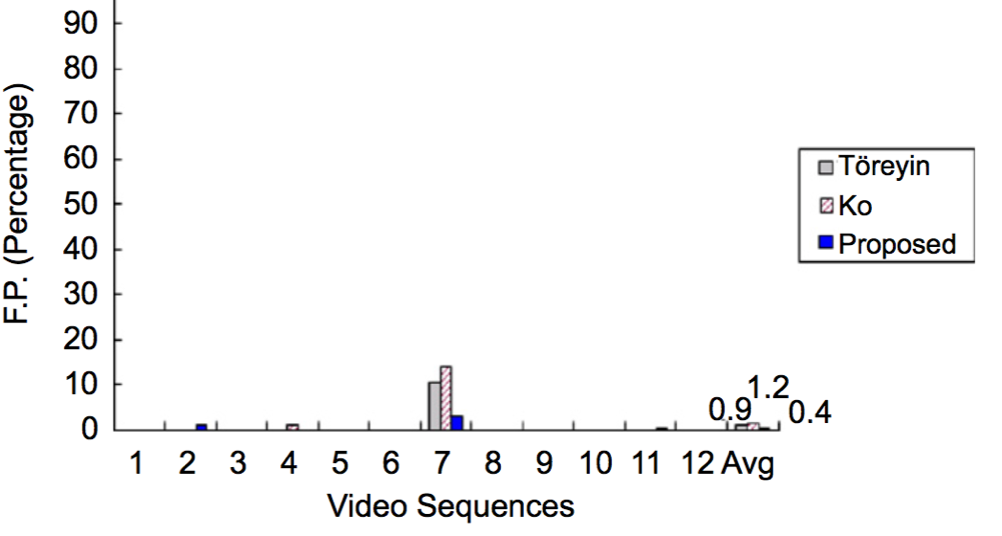


Figura 4-Bayesian Networks, comparação com outros métodos (Falsos positivos)

Por ultimo, foi feita a comparação em termos de rapidez, utilizando como medida os *FPS (Frames per secound)* em cada um dos 12 filmes. Como mostra a imagem seguinte.

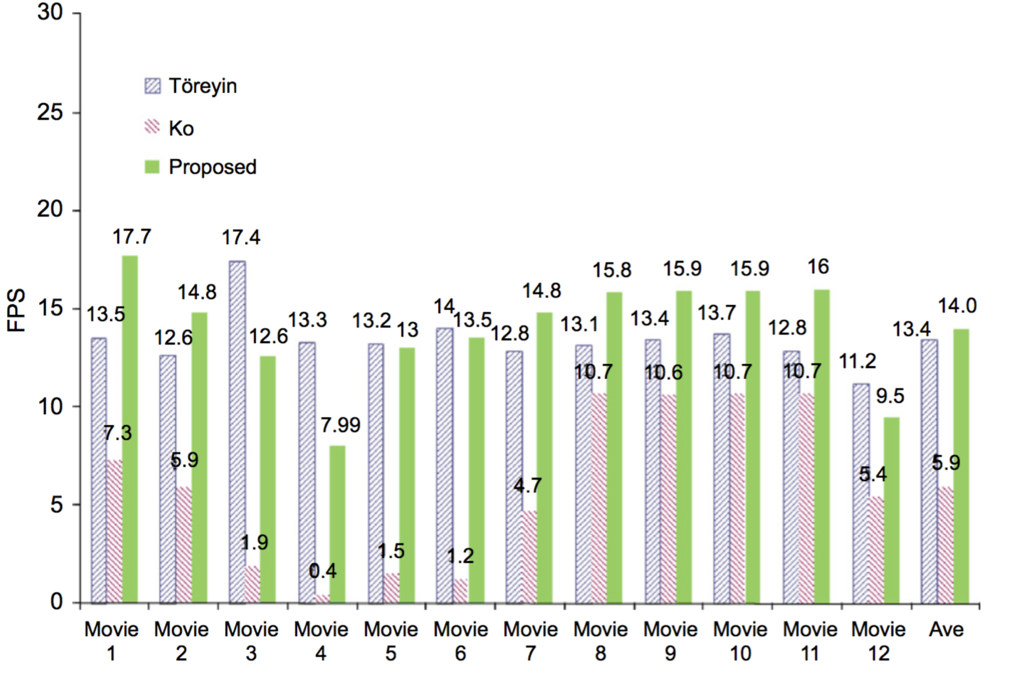


Figura 5-Bayesian Networks, comparação com outros métodos (Rapidez)

O anterior gráfico, contem alem dos 12 filmes de *dataset*, uma ultima coluna que mostra em média qual a quantidade de FPS que o algoritmo conseguiu processar.

### Método baseados em máquinas vetoriais

O sistema proposto [Máquinas vetoriais] propõe um sistema baseado em máquinas vetoriais. O sistema propõe resolver o problema por camaras de circuito fechado através da deteção pelas camaras e posterior envio por *SMS(Short message servisse)* e *streaming*. Na seguinte imagem podemos ver um diagrama da arquitetura geral da solução.

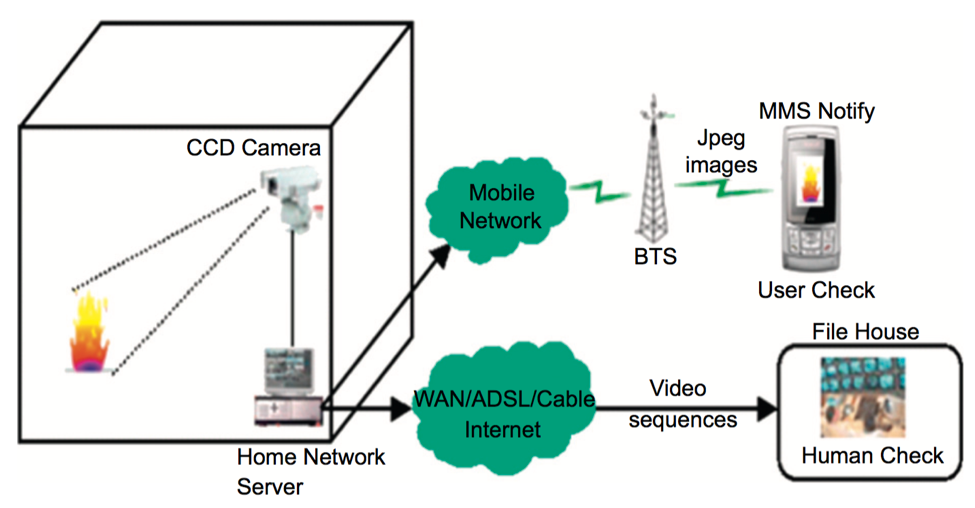


Figura 6-SVM, arquitetura da solução

A verificação é feita em vários passos como se pode ver na imagem seguinte.

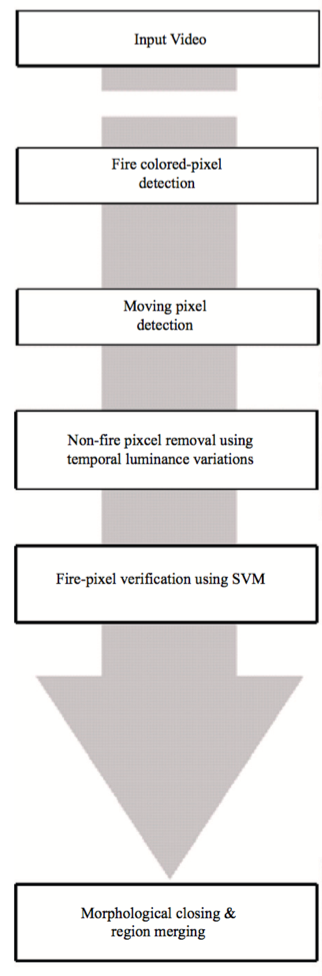


Figura 7-SVM, algoritmo

O primeiro passo é a tradicional deteção por pixéis de cores candidatas a pixéis de fogo. Esta deteção é feita com o pressuposto que uma região de fogo é mais clara que os vizinhos tanto em outdoor como em indoor. Para detetar os pixéis considerados como sendo fogo na imagem, o autor utiliza um mapa de cores em probabilidades uni modais de *Gauss*. Essas probabilidades são calculadas através de um *dataset* de imagens que contêm fogo. O pixel é depois comparado com a probabilidade e o algoritmo capaz de decidir se este é ou não fogo.

De seguida, é detetado o movimento com o objetivo de filtrar pixéis que não são fogo fazendo a diferença entre as localizações do pixel em dois *frames*. Se a diferença exceder um determinado *threshold* é considerado fogo, caso contrário, não é.

Apesar do sucesso na utilização dos dois métodos anteriores, é difícil para o algoritmo detetar objetos não fogo, mas com a mesma cor, e que se movem. Para suprimir a necessidade de ter um algoritmo que seja capaz de o fazer, foi utilizado um mapa de variações de luminosidade baseando-se no pressuposto que o fogo tem mais luminosidade. Para gerar o mapa, são utilizados 10 *frames* como *dataset*.

O ultimo passo, consiste na verificação por uma maquina vetorial ou *SVM(Support vector machine)*. Foram utilizados 12 vídeos como *dataset* para validação do sistema. Os vídeos estão descritos na seguinte imagem.

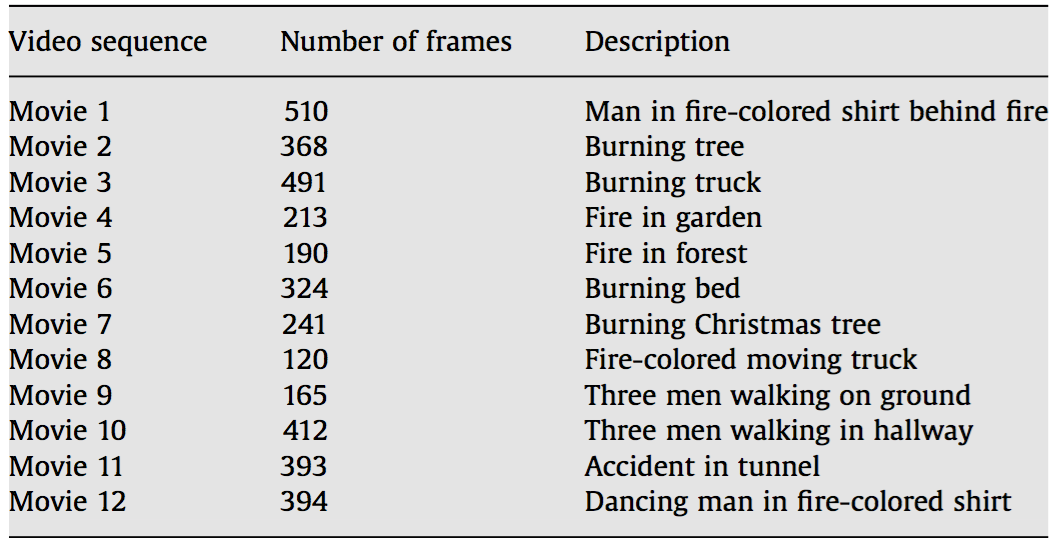


Figura 8-SVM, dataset

Este sistema é comparado com o método de *Toreyin*, o qual teve os resultados demonstrados na seguinte figura utilizando os mesmos vídeos.

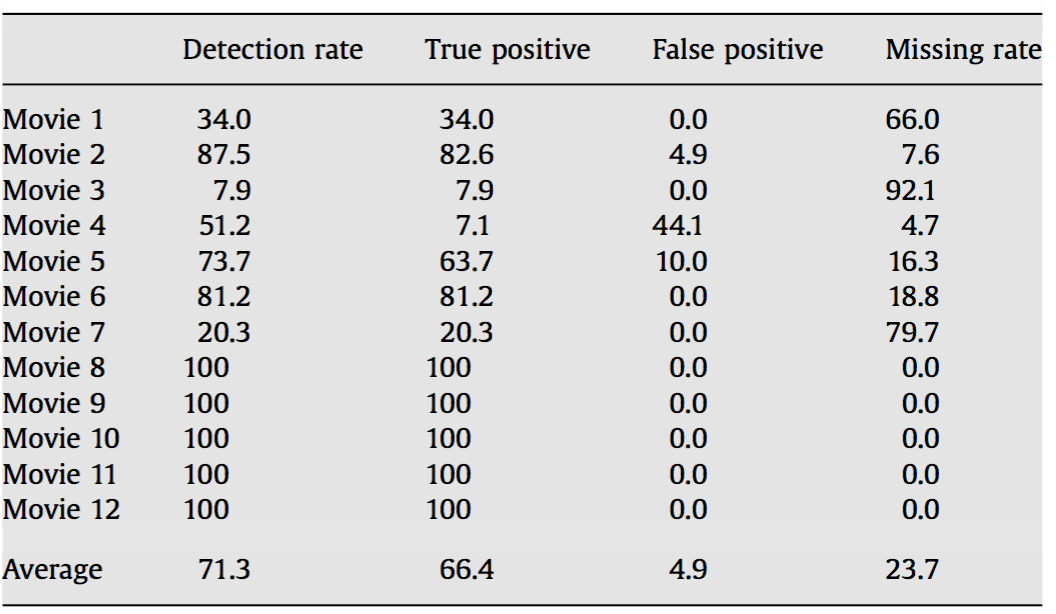


Figura 9-SVM, resultados do método de Toreyin

Os resultados conseguidos pelo sistema desenvolvido, mostraram-se bastante superiores como demonstrado na seguinte imagem.

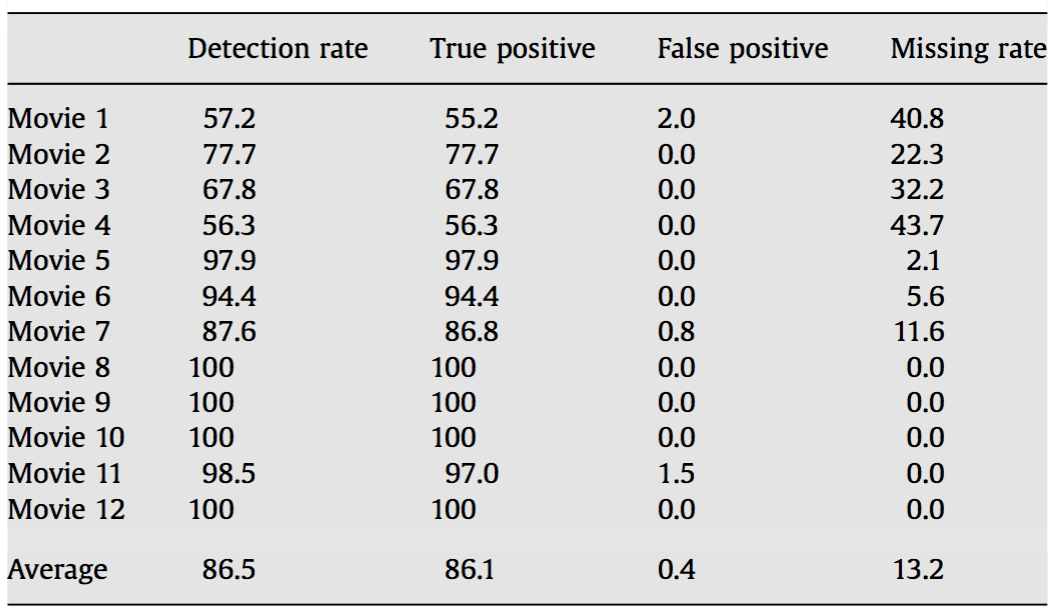


Figura 10-SVM, comparação com o método de Toreyin

### Métodos baseados em redes neuronais

O sistema proposto [Redes neuronais] propõe um sistema baseado em redes neuronais para resolver o problema da deteção de fogo. O método combina vários métodos de processamento de imagem para deteção de fogo e de fumo e uma rede neuronal *feedforward*.

Primeiro, é feita uma extração baseada na cor do fogo para eliminar interferências baseadas na cor. Depois são calculados os vetores da imagem com base na forma e na cor. De seguida, é utilizada uma rede neuronal para deteção da chama.

O autor faz o processamento da imagem baseando-se em dois conceitos. Características estáticas e dinâmicas da imagem. Para a categoria de característica estática, o autor utiliza a cor, forma e textura. Neste caso isolado, a cor está associada à temperatura. Com o aumento da temperatura a cor varia de vermelho saturado para menos saturado.

É retirado da imagem o vetor da textura através do *GLCM*. Por ultimo, como característica estática é retirado o vetor da forma baseado em formulas de Fourier. É ainda utilizado um vetor baseado na área da região, media e *centroid* da forma do fogo.

Como dinâmico é considerado o deslocamento da área no plano, a mudança de limites e a mudança de forma A mudança da área no plano supõe que a área do fogo muda muito no inicio do fogo. A área pode então ser calculada com o numero de pixéis com um valor superior a um *threshold*. Porem só este método pode levar a erro quando existem objetos não fogo que se movem. É também detetado o movimento geral do fogo através da análise de uma sequencia de imagens. É também utilizada a deteção da mudança dos limites. No inicio do fogo, os cantos das chamas mudam regularmente, ficando estável com o passar do tempo. O vetor da mudança dos limites na imagem vai ser retirado baseando-se na forma e curvatura do limite.

Existem dois processos. A segmentação da imagem que compreende a separação do objeto da imagem e identificação das partes necessárias para serem processadas. O autor utiliza uma segmentação baseada em *thresholds*. O segundo processo é o reconhecimento da imagem. Este processo pela análise, deteção, e reconhecimento da imagem contextualizando-a. A segmentação tem dois processos filho. A extração baseada em cor por o fogo ter uma cor bastante característica. Primeiro retira os valores *raw* do pixel da imagem. São depois convertidos par ao o espaço de cores RGB e analisado cada canal com base na formula da seguinte imagem.

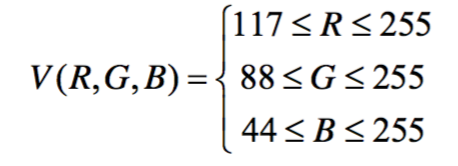


Figura 11-Redes neuronais, segmentação por cor

A segmentação do fogo da imagem utiliza a segmentação por regiões da imagem e métodos para segmentar o fogo em si.

A imagem é dividida em 25 regiões que são analisadas para ver a sua categoria. São depois juntadas as várias regiões da mesma categoria. De seguida o fogo é segmentado verificando todos os pixéis na mesma área até encontrar um pixel suspeito de ser fogo. Uma vez encontrado, são procurados pixéis vizinhos com intensidades semelhantes. O numero de pixéis encontrados é comparado a um *threshold*. Se for maior, pertence a área de fogo.

O ultimo passo é a decisão do reconhecimento sem interferência humana baseado numa rede neuronal. O autor, utiliza como input da rede neuronal os *feature* *vectores* extraídos nas fases anteriores de convergidos (cor e forma). Apesar de não existir um método formal para determinar a camada escondida (ou *hidden* *layer*) o autor apoia-se na formula da seguinte para encontrar o balanço entre o alto e baixo numero de neurónios na camada escondida.

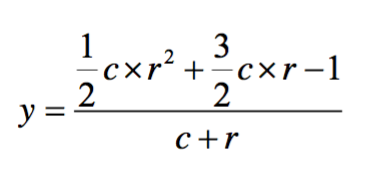


Figura 12-Redes neuronais, escolha do tamanho da camada oculta

A formula tem em conta o facto de quanto mais neurónios tiver a camada oculta da rede neuronal, maior o poder de processamento exigido ao hardware, maior o risco de *over*-*learning* e maior a facilidade de convergência do algoritmo de aprendizagem *back* *propagation*. A camada de output tem três neurónios. O vetor {1,0,0} de output significa uma cena de fogo, o vetor {0,1,0} significa que a imagem tem objetos de interferência. O vetor {0,0,1} simboliza a presença de floresta. A figure seguinte mostra o algoritmo global do sistema

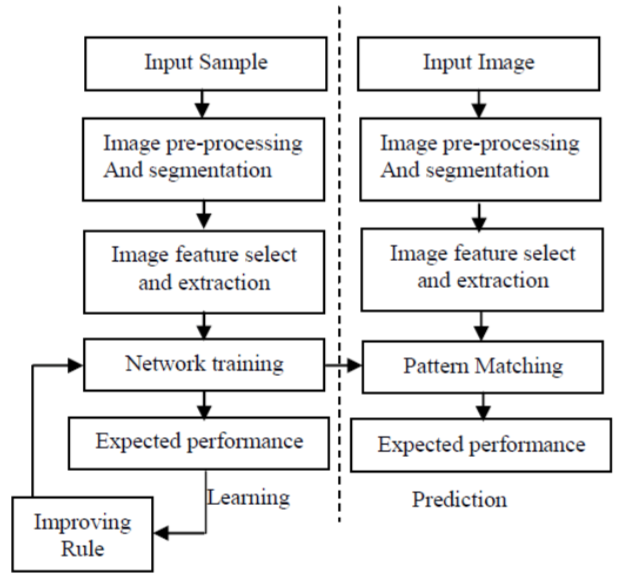


Figura 13-Redes neuronais, algoritmo

A rede neuronal, utiliza como algoritmo algoritmo *back* *propagation* baseado no ajuste de pesos e *bias* pelo *gradient* *descent*. A imagem seguinte mostra o treino da rede.

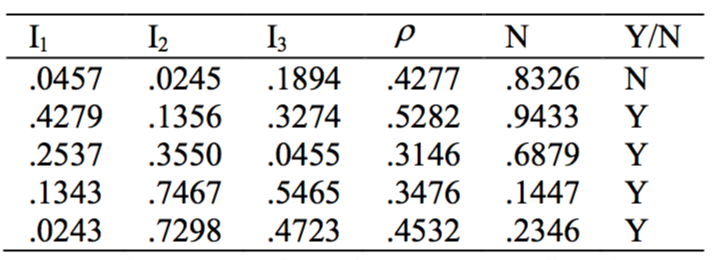


Figura 14-Redes neuronais, treino da rede

As primeiras três colunas, representam o os três canais de cores no espaço RGB, a quarta coluna representa a região suspeita de ser fogo. A quinta coluna representa o numero de cantos da fama. A próxima imagem ilustra o resultado dos testes.

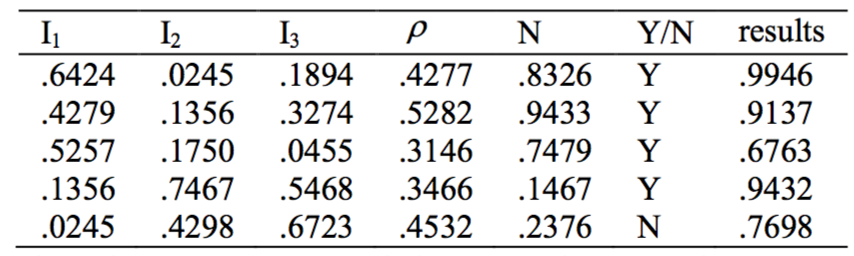


Figura 15-Redes neuronais, resultados dos testes

O autor escolheu após o treino, um *threshold* de 0.8. Um resultado menor que 0.8 indica não fogo, maior indica fogo. O autor reporta um *recall* de 75%.

## Sistemas comerciais

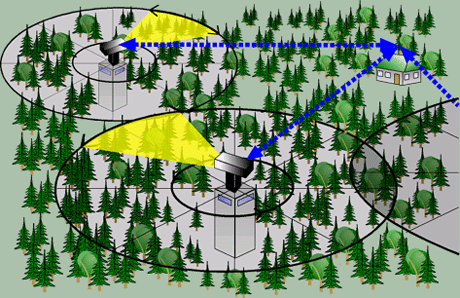
Neste subcapítulo são apresentados alguns sistemas comerciais encontrados no decorrer da investigação do projeto.

### Firewatch

É um sistema terrestre de vigilância remota de incêndios na floresta que emite alarmes quando o fogo é detetado. Este sistema está atualmente em utilização na Alemanha em vários estados e em várias cidades da Europa. O sistema conta com as seguintes características:

* Deteção através de uma camara montada numa torre;
* Deteção de nuvens de fumo de dia e de noite;
* Transmissão de dados via cabo ou rádio;
* Supervisão rápida de uma área de 70.000 hectares;
* Elevada qualidade de imagem;

A seguinte imagem mostra a arquitetura da solução FireWatch.



Cada sensor ótico, é capaz de rodar 360º num espaço de 4 a 10 minutos com cada rotação a representar um ângulo de 10º. A cada posição, recolhe 3 imagens. Este sistema assenta na supervisão humana, não se preocupando com a taxa de erro do sistema.

### ZeroFires

O ZeroFires contem três soluções de combates a incêndios, sendo que uma se baseia em software e duas em hardware.

A solução ***ZF-WildFire*** station series é uma solução de deteção de fogos florestais desenhada para ser implementada em torres de vigia de múltiplas maneiras, incluindo em postes. A solução implementa um software com capacidade de analisar a imagem e procurar na mesma fogo ou fumo. Uma vez detetado um dos elementos é enviado um alarme (Web, e-mail ou SMS) por dados móveis, ou pelo sistema integrado de comunicações por satélite. O sistema é capaz de detetar fogo de 1 metro quadrado e cada camara consegue cobrir 12.500 km quadrados. Em relação ao sistema anterior este é claramente menos abrangente. Mas em questão de rapidez, este é capaz de processar 2000 km quadrados por minuto. Estas estações são alimentadas por painéis solares e energia solar. A seguinte imagem, mostra a instalação deste sistema.



Figura 16-ZF-WildFire station series

A segunda solução chamada ***ZF-Portable system*** é semelhante à anterior solução, mas esta é portátil. A solução foi desenvolvida para poder ser facilmente transportável. Em termos de características técnicas, é capaz de detectar fogos de 1 metro quadrado num espaço de 18 km quadrados ou fogos de 12 metros quadrados num espaço de 12 km quadrados. A seguinte imagem, mostra a solução descrita.



Figura 17-ZF-Portable system

A terceira solução chamada de ***ZF-Dashboard*** é uma solução baseada na comunicação e interface para o utilizador com os sistemas anteriormente descritos. Este software permite a comunicação com as camaras e definição de parâmetros como alarmes, velocidade da camara entre outros. Este software está adaptado para dispositivos móveis e desktop. A próxima imagem ilustra a aplicação.



Figura 18-ZF-Dashboard

# Técnologias e àreas estudadas

Neste capitulo, são descritas as tecnologias estudadas no desenvolvimento do projeto. Como já referido, o trabalho descrito no presente relatório assenta sobre o tema visão computacional. Esse tema é dividido em duas grandes áreas. O processamento de imagem, e a aprendizagem máquina.

## Processamento de imagem

O processamento de imagem é o método de conseguir através de operações sobre a mesma, extrair desta informação útil. Neste tipo de técnica, por norma, o input é uma imagem, e o output são as características extraídas.

O processamento de imagem inclui tipicamente os três passos seguintes:

* Aquisição da imagem;
* Análise e manipulação da imagem;
* Output do processamento;

O problema que o processamento de imagem tenta resolver é o representar de uma imagem e extração do mesmo tipo de informação que o ser humano consegue ter retirar. Para o ser humano é trivial distinguir por exemplo que a imagem seguinte tem dois logotipos e um corresponde ao logotipo da biblioteca *openCv* e o outro da linguagem de programação *python*.

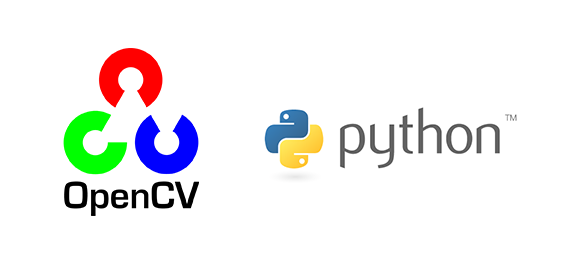


Figura 19-Logotipo openCv e python

O computador, ao analisar uma imagem, não tem perceção do que representam os conteúdos visuais. A camara capta a reflexão da luz sobre determinado objeto e representa a imagem sobre a forma de três matrizes de inteiros de 8 bits por cada posição. Essas três matrizes representam um canal de um esquema de cor. Por exemplo, caso seja utilizado o espaço de cores *RGB*(RedGreenBlue) cada uma das matrizes representa respetivamente o *red*, *green*, e *blue*. Existem, no entanto, outros que tentam imitar melhor a cor e ser mais intuitivos como opor exemplo o *HSV(HueSaturationValue)* entre outros.

Cada pixel representa um valor de 0 a 255, onde 0 é a ausência de cor e 255 é o branco. Por exemplo, com o vetor numa posição (x,y):

* O vetor [0,0,0] seria um preto;
* O vetor [255,255,255] seria um branco;
* O vetor [255,0,0] seria um vermelho;
* O vetor [0,255,0] seria um verde;
* O vetor [0,0,255] seria um azul;

A imagem seguinte mostra como é vista uma imagem de 5\*5 pixéis do ponto de vista do computador em que cada matriz representa um dos três canais *RGB*..

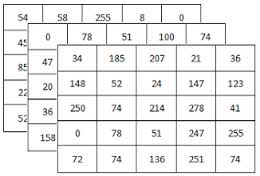


Figura 20-Imagem do ponto do vista do computador

### OpenCv

O processamento de imagem, como referido no ponto 3.1, compreende operações entre matrizes para a extração de indicadores. Essas operações são extremamente intensivas em termos matemáticos e como tal, é muito fácil escalar ao ponto de não se conseguir obter os melhores resultados e ter um compromisso elevado com a performance.

Para colmatar esse entrave, entre várias, foi desenvolvida a biblioteca *OpenCv*. Esta Framework contem de uma forma muito organizada os principais métodos de processamento de imagem onde se destacam os seguintes:

* Operações básicas como a leitura, escrita e visualização de imagens;
* Conversão entre espaços de cor;
* Acesso fácil a todas as características da imagem como cor de um pixel, regiões, forma entre outros;
* *Gradients*, *kernels*, *smoothing* e *blurring*, deteção de limites, *contours* e histogramas;
* *Canny* *edges*;
* Redimensionamento, *thresholding*, escala entre outros;
* Operações aritméticas e morfológicas entre imagens;
* Image *Descriptors*;

Existem outros métodos que se podem ver na documentação da biblioteca. Também para ajudar nas operações algébricas, é muitas vezes utilizada a biblioteca *numpy*, No âmbito deste projeto, ambas as bibliotecas são utilizadas.

## Aprendizagem máquina

O problema dos fogos, representa um problema de classificação. O principal requisito foi que o sistema fosse capaz de distinguir entre um cenário com fogo e um cenário sem fogo. Entre outras técnicas, são utilizados para a resolução deste tipo de problemas métodos de *machine learning* ou aprendizagem máquina. Esta técnica permite através da implementação de algoritmos fazer o computador aprender sobre um tema sem ser explicitamente programado para o fazer. Existem dois tipos de aprendizagem. *Supervised learning* ou aprendizagem supervisionada, e *unsupervised learning* ou aprendizagem não supervisionada.

**A aprendizagem supervisionada**, implica o conhecimento prévio dos elementos utilizados para o treino do sistema através de *labels* ou categorias de cada um dos elementos. Num exemplo prático, se quiséssemos reconhecer na imagem uma cadeira ou uma mesa, esses dados seriam dados como input ao algoritmo juntamente com a respetiva descrição. Dessa forma, o algoritmo aprende com conhecimento prévio o que significa cada imagem. As imagens passadas ou qualquer outro tipo de dado para aprendizagem é chamado de dataset. O conceito de *dataset* é explicado nos próximos parágrafos.

**A aprendizagem não supervisionada** é utilizada para agrupar em categoria elementos não conhecidos. Sintetizando e como exemplo damos um *dataset* sem descrição em que existem duas categorias de conteúdos fotográficos que nós sabemos. Cem fotografias de lagos, e duzentas fotografias de campos agrícolas. Neste cenário, é possível desenvolver um algoritmo que sem qualquer conhecimento consiga classificar e agrupar em duas categorias cada uma das imagens.

Antes da implementação do algoritmo é necessário reunir um *dataset* com exemplos que que queremos que o programa aprenda. Este *dataset* deverá ser grande e o mais diversificado possível para aumentar a precisão do algoritmo. Após reunido o *dataset* é necessário proceder à sua divisão para passar pelas até três fases deste tipo de sistemas.

**A primeira fase é a fase de treino**. Nesta fase o utilizador de todo o *dataset*, seleciona uma percentagem que vai utilizar para o algoritmo de *machine learning* aprender.

**A segunda fase, e esta é uma fase opcional, é a fase de validação**. Esta fase é utilizada dependendo do algoritmo a utilizar. Os algoritmos contam com parâmetros que alteram a sua performance. Estes parâmetros são chamados de *hyperparameters*. Estes *hyperparameters* são por vezes um pouco aleatórios e só são conseguidos através de tentativa erro. Para selecionar os melhores *hyperparameters* é utilizada a fase de validação. Esta fase utiliza uma pequena percentagem do *dataset* de treino (tipicamente 5 ou 10%) para conseguir testar os vários *hyperparameters* e escolher os melhores.

**A ultima fase, é a fase em que é validada a aprendizagem.** Esta fase chama-se de fase de teste. Nesta fase é selecionada uma percentagem não maior que 33% do *dataset* que é utilizada o algoritmo prever os valores de input consoante a aprendizagem feita na primeira fase. Na seguinte imagem podemos ver valores típicos utilizados para a divisão do dataset.

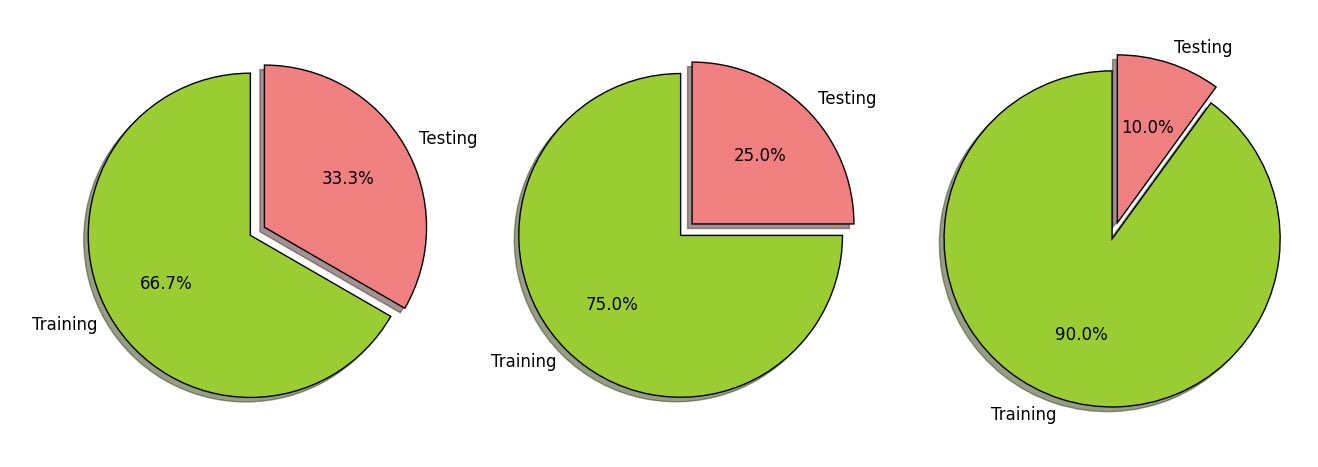


Figura 21-Divisão do dataset

Em qualquer um dos casos, é extremamente importante que o *dataset* ao ser dividido seja isolando, evitando por exemplo elementos de treino na fase de teste. Caso isso acontecesse, estaríamos a treinar o algoritmo para passar na fase de teste.

A aprendizagem máquina é uma área muito vasta que implica muito estudo para ser dominada. Existem atualmente vários algorimos que poderiam com diferenças de performance solucionar o problema no presente relatório exposto.

### Support Vector Machines

*SVM*´s ou *support vector machines* é um dos temas em *machine learning* sobre o qual se pode escrever por horas e ter muito a dizer. O algoritmo é extensivamente matemático, e é fácil nos perdermos no estudo do mesmo.

A razão das *SVM*´s serem tão populares, é que é possível ter um bom classificador sem perdermos muito tempo a afinar os *hyperparameters*.

Para entender como funcionam as *SVM*´s é necessário entender primeiro o conceito de separação linear. A separação linear acontece quando, num pár com dois conjuntos, é possível traçar uma reta e separar os dois como ilustra a seguinte imagem.

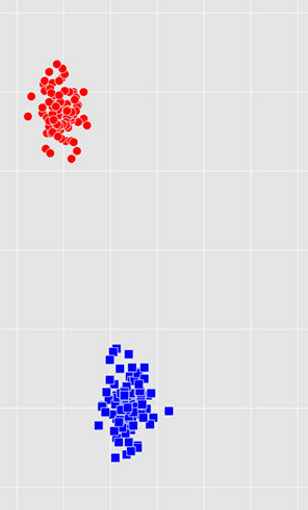


Figura 22-Separação Linear possivel

Neste caso podemos claramente separar as duas classes com uma reta. Um exemplo de classes que não são linearmente separáveis é ilustrado na seguinte figura.

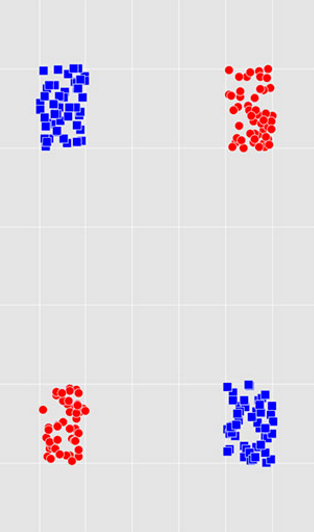


Figura 23-Separação linear impossivel

É impossível, com uma só reta separar estas quatro classes. Podíamos eventualmente traçar uma segunda reta. Reta essa que iria sobrepor a primeira, o que também torna as classes linearmente não separáveis. Este é conhecido como o problema do OR exclusivo.

A linha utilizada para separar as classes, em duas dimensões é somente uma linha, em três, é um plano, e em mais que três dimensões é um *hyperplane*. Esta é a linha fronteira utilizada para classificar as classes. Quanto mais afastada uma classe estiver da linha, maior a confiança do algoritmo na classificação. A razão pela qual isto acontece é porque, existe uma maior margem de confiança o que dá uma maior flexibilidade quando classificamos os dados para fase de teste como demonstra a figura seguinte.

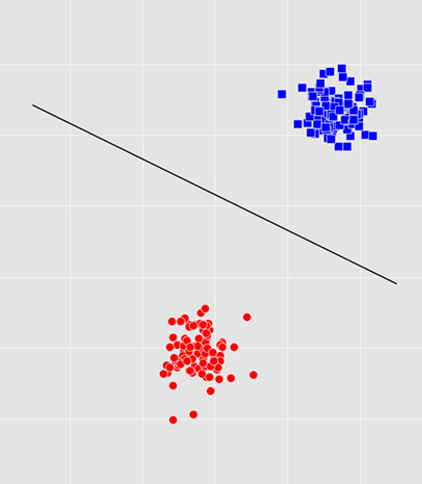


Figura 24-Linha fronteira

A linha de fronteira é decidida com base nos *support* *vectors*, daí vem o nome do algoritmo. Os *support* *vectors* são os vetores de cada classe que se encontram o mais próximo da linha fronteira como mostram os elementos da próxima imagem.

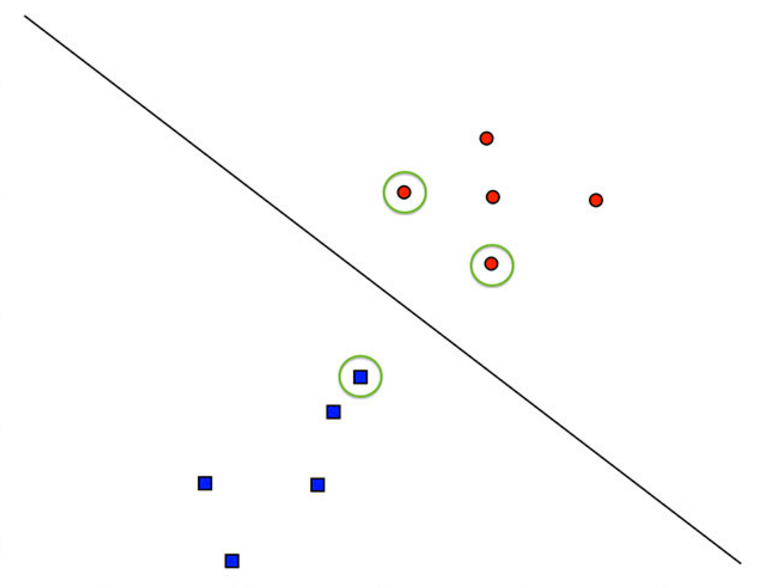


Figura 25-Support Vectors

Com esses *support* *vectors*, é maximizada a margem da linha fronteira separando as classes de maneira ótima.

Uma parte do problema está endereçada. Como conseguir duas classes serem classificadas com boa confiança. A outra separar linearmente várias classes. Sem esse problema resolvido, o algoritmo não passaria de um classificador linear.

A ideia para a separação nesses casos é. Uma vez que não são separáveis no plano de duas dimensões, são projetados em mais de duas dimensões e calculado um *kernel*. Este *kernel*, é o *kernel* que vai otimizar a separação quando existem mais de duas classes.

Seguem os quatro tipos de *kernel* mais utilizados:

* Linear;
* Polinomial;
* Sigmoid;
* RBF;

O *kernel* a utilizar é muito importante no sentido que, cada *kernel* tem as suas especificidades. Por exemplo. O *kernel* linear é representado pela seguinte função:

Podemos ver que a função só tem como *hyperparameter* o T. Vendo a função polinomial.

Vemos que a função polinomial tem mais dois *hyperparameters* para afinar.

Concluindo. O *SVM* é um algoritmo de *machine learning* muito poderoso, difícil de entender, mas fácil de utilizar. Dependendo do *kernel* escolhido, a fase de validação do algoritmo pode ser mais ou menos demorada.

Este é um algoritmo que é muito rápido a treinar mas lento a validar.

### Decision trees e Random Forests

As *Decision* *trees* e *Random* *Forests* ambas técnicas de *machine learning* diferentes mas que estão de alguma forma interligadas.

Às *decision* *trees* ou arvores de decisões é dado uma imagem de treino, extraído um *feature* *vector*, e criados nós de decisão. Ao contrário de outras técnicas, esta tem a vantagem de permitir a cada nível da arvore verificar o comportamento com a imagem facilitando por vezes o trabalho do programador. A seguinte figura mostra um exemplo de uma arvore de decisão.

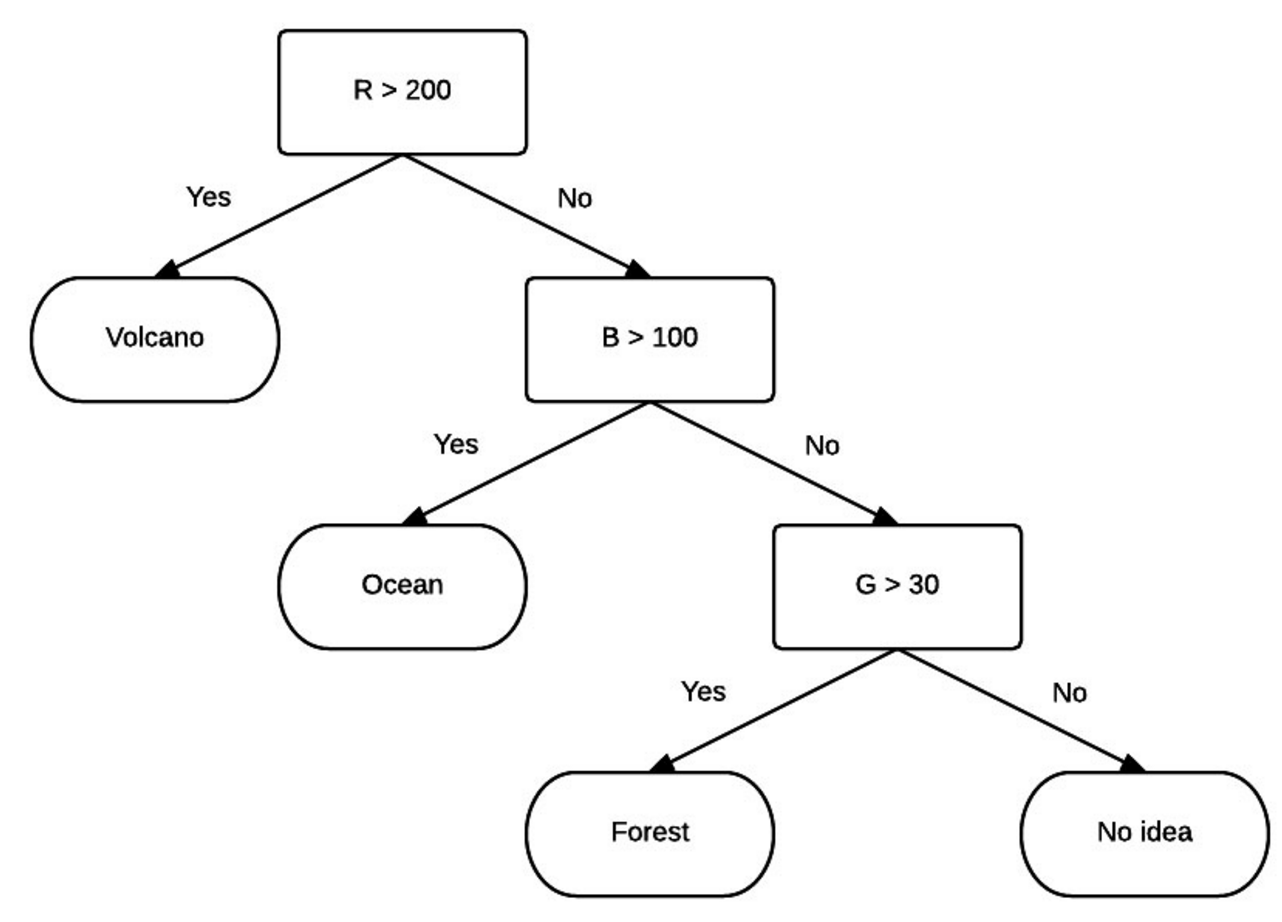


Figura 26-Arvore de decisão RGB

No exemplo da imagem anterior, a arvore de decisão foi construída com base na média dos valores RGB da imagem de input. Como exemplo, uma imagem que extraísse o seguinte *feature* *vector*.

Esta imagem seria classificada de acordo com a arvore como sendo uma floresta. Estes *feature* *vectors* podem ser extraídos com vários algoritmos que constroem automaticamente a arvore através dos dados de treino.

O algoritmo utiliza o que se chama de *informative* *split* para depois de ter os *feature* *vectors* certos conseguir a melhor divisão da arvore tentando todas as *features* e vendo qual funciona melhor.

As *Random* *Forests* começam onde as arvores de decisão acabam. Alguém, após fazer várias arvores de decisão, decidiu construir uma arvore de arvores de decisão. As *Random* *Forests* encaixam na categoria de *ensemble* *methods*. Em vez de utilizarem um só classificador, utilizam muitos como mostra a imagem seguinte.

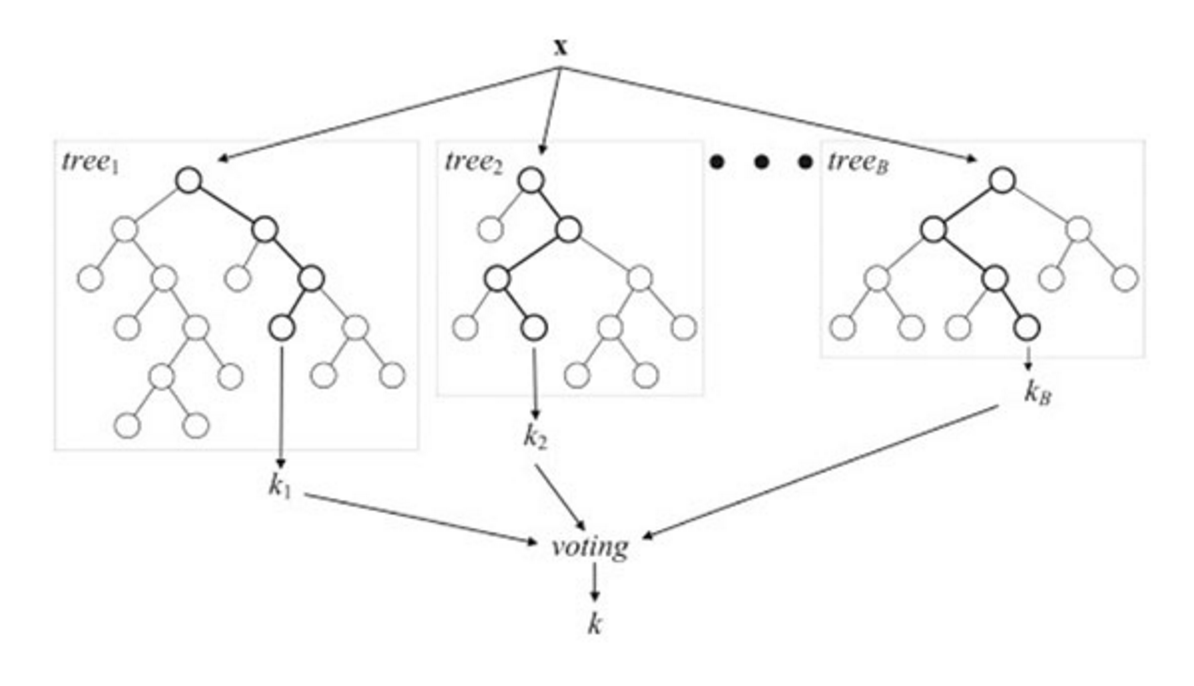


Figura 27-Random forest

Cada arvore de decisão, vota para uma decisão final. A categoria com mais votos, ganha. Os *ensemble* *methods* são baseados na inequalidade de Jensen, que diz que um método ensemble com os resultados combinados e feita uma média, tem menos erro que a média de resultados de um só método.

O método alem de agregar vários resultados, adiciona também alguma aleatoriedade para o treino de cada arvore. Este passo é chamado de ***bootstrapping***. De uma forma resumida, cada arvore de decisão vai contar com o numero de nós igual à metade das imagens de treino em que os *feature* *vectors* dessas imagens participam na construção da arvore. O **segundo passo** é atribuir mais um grau de aleatoriedade ao sistema. Supondo que temos um *feature* *vector* com dimensão de 128. Ao 128 aplicamos a raiz quadrada ou o logaritmo. O resultado dessa operação dita um novo *feature* *vector* para cada imagem. O passo final é o voto de cada arvore.

O algoritmo de *Random* *Forest*, é chamado em *machine learning* como um método de *ensemble* learning o que será descrito no próximo subcapítulo.

### Aprendizagem Ensenble

O *ensemble* *learning* combina resultados de vários modelos para ter melhores resultados. O *Random* *Forest* é um exemplo de *ensemble* *learning* pois combina o resultado de várias arvores de decisões.

O *ensemble* *learning*, quando bem aplicada, supera em performance qualquer algoritmo de *machine* *learning*. Existem vários métodos de *ensemble* *learning* entre os quais os mais comuns:

* **Bagging** - Com um conjunto aleatório de exemplos dos dados de treino damos para diferentes versões do modelo. No exemplo do *random* *forest*, para diferentes arvores de decisão. e deixamos que votem na decisão final
* **Boosting** – Com esta técnica identificamos quais os pontos fracos do *dataset* e tenta torna-los fortes pegando nos pontos fracos identificados na fase de treino e passando-os a outras fases do algoritmo.
* **Bucket os models** – Este método tem em conta vários modelos sendo estes iguais ou diferentes e escolhe o melhor modelo
* **Stacking** – Igual ao modelo *Bucket* *of* *models*, mas em vez de escolher o melhor modelo, combina todos os resultados. Este é o exemplo do *random* *forest*.

### Convolutional Neural Netwoks

Os métodos anteriores, apesar de serem métodos de *machine* *learning*, são em tudo diferentes das redes neuronais convulsionais no sentido em que os anteriores métodos se apoiam fortemente em matemática ao contrário deste.

Para falar em redes neuronais convulsionais temos de falar em **redes neuronais artificiais**. As redes neuronais artificiais foram desenvolvidas em analogia com a maior máquina de processamento conhecida. O cérebro! O cérebro contem vários neurónios que se interligam entre eles e transmitem impulsos elétricos. Esses neurónios “assimilam” conhecimento e estão constantemente em aprendizagem desde o momento em que nascemos até ao momento em que morremos. Para termos uma ideia do tamanho do cérebro, este é dividido em vários córtices. O córtex primário ou como é conhecido o V1, contem cerca de 140 milhões de neurónios para aprendizagem de conteúdo visual com biliões de ligações entre eles. Mais complexo fica, quando estudamos e constatamos que o córtex V2,V3, V4 eV5 também participam no reconhecimento da imagem.

As redes neuronais artificiais foram desenvolvidas com apoio na filosofia do cérebro sendo o elemento mais básico da rede o neurónio. O primeiro tipo de rede a ser desenvolvido, foi uma rede de *perceptrons* o que deu origem às *Multi*-*layer* *Perceptron* *Networks* ou *MLP´s*. O *perceptron* na sua forma mais básica, tem vários inputs e pode fornecer um ou mais outputs. Para compreendermos o exemplo seguinte demonstra um *perceptron* com inputs x1 e x2.

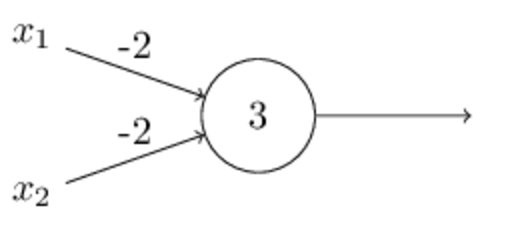


Figura 28-Perceptron

Os inputs do perceptron são depois ponderamos com um peso (-2) e enviados para o neurónio. O neurónio tem implícito um valor chamado Bias (3). Concluindo, o resultado de um *perceptron* é o somatório da multiplicação dos inputs com os pesos da ligação e soma do *bias* do neurónio.

O *bias* é implesmente um valor que indica o quão fácil é o neurónio disparar a 1 ou a 0. Depois de disparado, e caso seja a ultima camada ou camada de output da rede neuronal, o resultado passa por uma função de ativação que vai indicar o valor final da rede. A imagem seguinte mostra o funcionamento simples da rede neuronal do input ao output a mais baixo nível mostrando a função de ativação do *perceptron*.

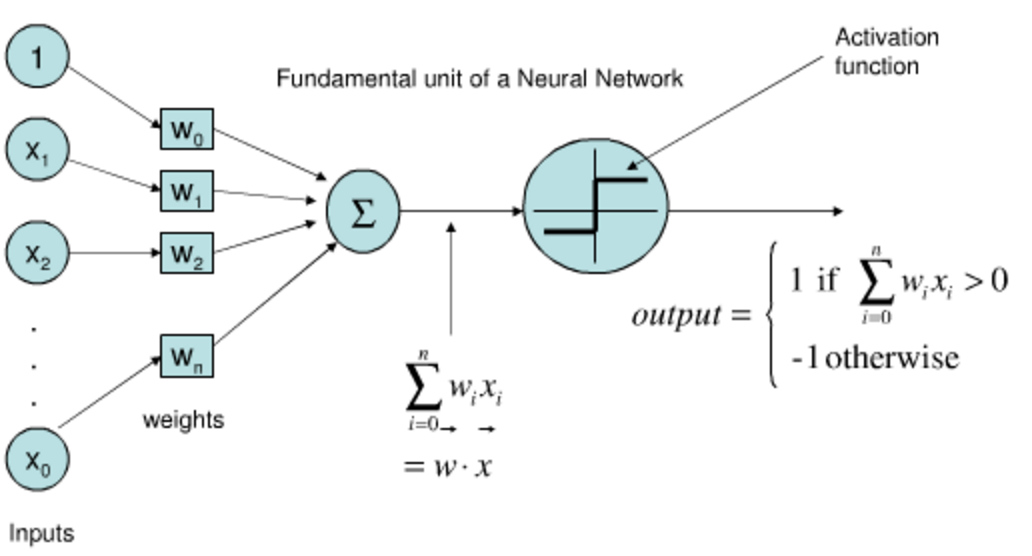


Figura 29-Arquitetura simples de rede neuronal

Esta função de ativação chamasse de função *Step function* ou função de passo que será explicada nos próximos parágrafos.

Existem ainda dois conceitos a serem explorados. O conceito de *layers* e o neurónio do tipo *sigmoid*.

A rede neuronal, seria extremamente simples se contasse apenas com um neurónio e também extremamente ineficaz no que toca a problemas complexos. Na imagem 30 ilustramos uma porta NAND que é resolvida com um *perceptron*. Mas se quiséssemos por exemplo adicionar dois bits com portas NAND, não conseguiríamos com aquele modelo. A seguinte imagem ilustra quais seriam os requisitos do problema.

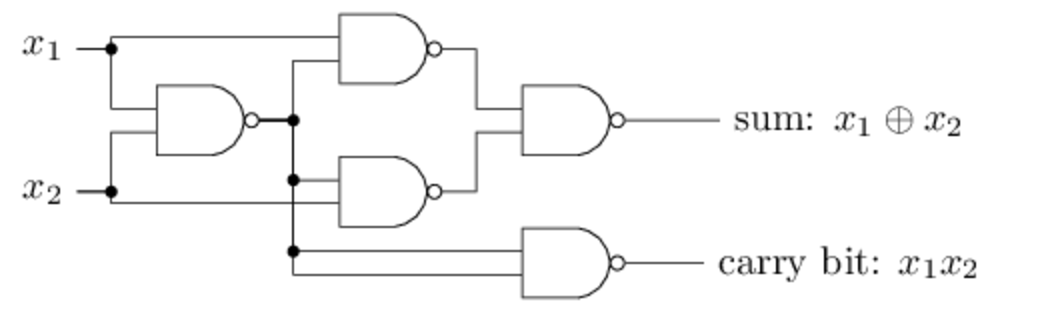


Figura 30-Soma de dois bits com NAND´s - Ponto de vista digital

Cada *perceptron* só consegue neste caso modelar uma porta. Nesse sentido facilmente nos deparamos com um problema. Para o resolver, as redes neuronais contam com não um, mas vários neurónios divididos em várias camadas.

* **A camada de input**. Esta é a primeira camada que recebe os inputs;
* **A camada oculta**. Esta camada é onde é feita toda a computação. Multiplicados os pesos e adicionados os *bias*. Podem existir várias camadas ocultas;
* **A camada de output**. Esta é a camada responsável por dar um resultado da rede;

A seguinte imagem mostra uma rede neuronal das várias camadas. Esta rede resolve o problema de adição de dois bits com portas NAND.

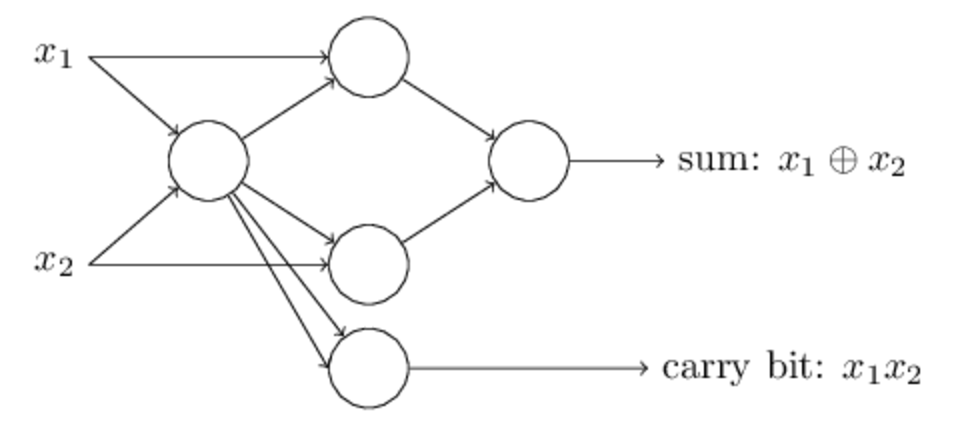


Figura 31-Soma de dois bits com NAND´s - Ponto de vista neuronal

A próxima evolução, foi a introdução da função *sigmoid*. A função do *perceptron* é como representa a seguinte imagem.

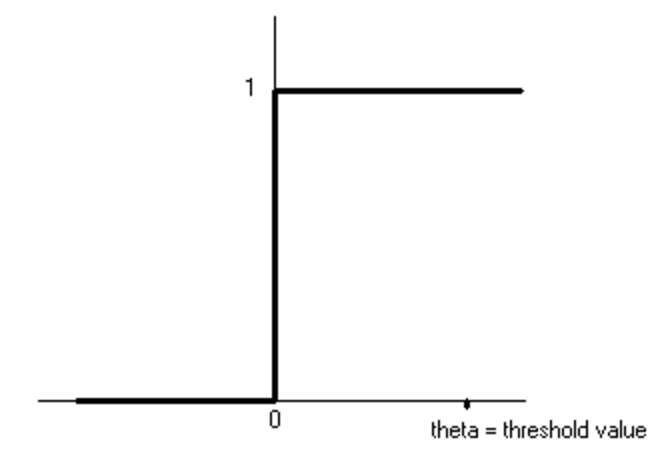


Figura 32-Função de passo

Esta função introduz um problema na rede que é comum com os algoritmos de *machine* *learning*. A aprendizagem. Com esta função não é possível atribuir um grau suficientemente bom de aprendizagem à rede. Alias, com esta função de ativação, uma pequena alteração num *bias* ou num peso pode alterar todos os resultados da rede radicalmente uma vez que o seu output é binário.. Por esse motivo foi introduzida a função *sigmoid* para função de ativação da rede neuronal. A função é representada no gráfico seguinte.



Figura 33-Função sigmoid

Esta função introduz uma maior capacidade de aprendizagem à rede. O gráfico é *representado* pela seguinte função.

Em que o valor –z é o resultado do neurónio. A seguinte imagem mostra uma rede neuronal utilizada para classificar imagens do *dataset* de MNIST.

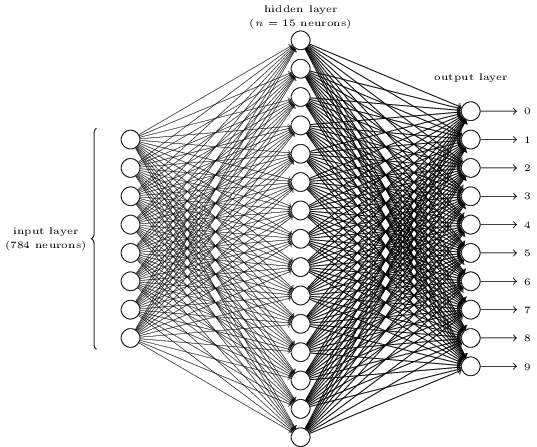


Figura 34-Rede neural

O numero de neurónios de input (784) é dado neste caso pelo numero de pixéis da imagem e o numero de neurónios de output é dado pelos resultados expectados da rede. Numero de 0 a 9.

As redes neuronais convulsionais são semelhantes às redes neuronais convencionais no sentido em que utilizam pesos e *bias* passiveis de serem aprendidos. Porem, as redes neuronais ditas normais, não são escaláveis para imagens completas. No exemplo anterior vimos que a camada de input tinha 784 neurónios. Esse valor foi conseguido pela multiplicação de 28 por 28 pixéis da imagem de input numa escala de cinzentos. Se tivermos a mesma imagem na escala RGB teríamos de ter 28\*28\*3 neurónios na camada de input o que indica que esta estrutura não é escalável a grandes imagens e poderia ficar bastante complexa. Não só, quando maior o numero de neurónios de input, significa que mais características a rede vai ter de aprender. Essas características são aprendidas na camada oculta. No exemplo anterior podemos fazer a analogia de cada neurónio da camada oculta como uma extração de uma característica da imagem a alto nível. Muito comum é haver a necessidade de extrair detalhes da imagem. Para isso seria necessárias mais camadas ocultas que com o aumentar do nível da camada, aumentava também a abstração da imagem em si. Cada nível de camada oculta, extrairia uma característica da característica extraída no nível anterior. Várias camadas ocultas traria um problema grave para a aprendizagem da rede. O *gradient* que é o valor do erro do neurónio, tenderia a perder-se implicando que os neurónios mais perto da camada de input poderiam não aprender muito pouco ou nada.

As redes convulsionais foram desenvolvidas para ter em contra o processamento de imagem. Ao contrário das redes neuronais convencionais, a rede convulsional conta com camadas isoladas de três dimensões. Cada camada contem uma altura, uma largura e uma profundidade em que a profundidade representa o numero de canais. A seguinte imagem compara as duas arquiteturas.

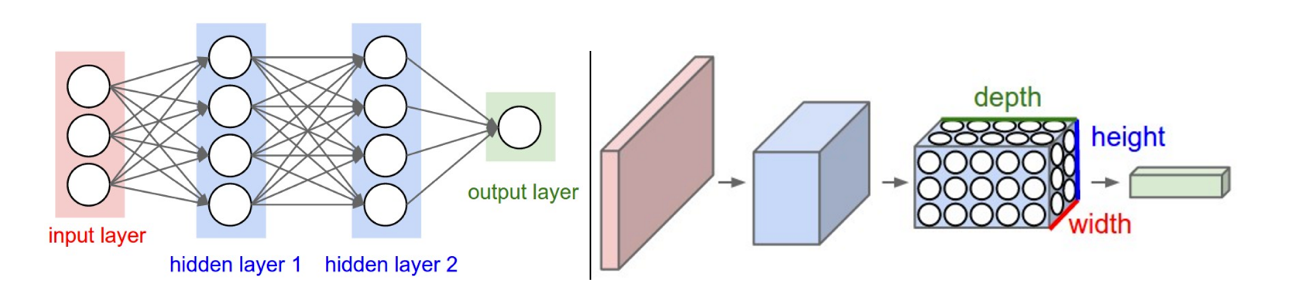


Figura 35-Rede neuronal VS rede convulsional

Também a maneira de ligar as diferentes camadas é diferente nas redes convulsionais. Ao contrário das redes convencionais, estas ligam os neurónios de uma camada a só uma pequena parte dos da camada anterior. Como camada final, a rede convulsional apresenta um vetor organizado somente na dimensão de profundidade.

Uma rede convulsional ou *CNN* tem uma lista de camadas que transformam a imagem de input numa imagem diferente através de uma função diferenciável. Existem algumas camadas diferentes:

* **Camada convulsional** – Consiste em vários filtros. A cada input, é feita uma convulsão produzindo um mapa do filtro convulsionado com a imagem
* **Pool e Relu** – Camadas inseridas entre camadas convulsionais. O objetivo é controlar o *overfitting* e reduzir o numero de parâmetros e consequentemente de computação da rede. Também é obvio A camada de Pool entre outas coisas consiste num *kernel* de tamanho dois por dois que maximiza os pixéis e faz a convulsão para um novo tamanho. Assim consegue passar de uma imagem de 224 por 224 pixéis para 112 por 112 pixéis
* **Fully Connected** – Os neurónios nesta camada têm ligações a todos os neurónios das camadas anteriores como acontece nas redes neuronais convencionais que depois produzem um output consoante a função de ativação

A seguinte imagem mostra o processamento de uma imagem pela rede convulsional.



Figura 36-Demonstração das camadas da CNN

Podemos ver pela imagem, as características recolhidas por cada camada da rede neuronal convulsional.

## Energias renováveis

Foram também estudadas, de modo a tornar o sistema o mais autossustentável possível dois métodos de extração de energia. A energia solar e a energia eólica. Nos próximos dois capítulos são descritos os métodos de extração deste tipo de fontes de energias renováveis.

### Energia solar

A energia solar, provem do sol. O sol é capaz de gerar energia suficiente para alimentar todo o mundo e muito mais que pudéssemos utilizar. Para converter a luz e e calor fornecidos pelo sol em energia, são utilizadas células solares.

As células solares são dispositivos elétricos que recolhem a luz solar e a transformam em energia. As células solares são muito pequeninas com forma octogonal que juntas foram módulos de células solares, que por sua vez se agrupam formando painéis solares.

Estas células, são células normais como as células de uma bateria de um carro, mas em vez de produzirem energia através de produtos químicos, produzem através da luz solar. Razão pela qual são muitas vezes chamadas de células fotovoltaicas. A imagem seguinte podemos ver a representação de um painel solar.



Figura 37-Painel Solar

### Energia eólica

Outra forma de energia renovável, é a energia eólica. A energia eólica é um tipo de energia que é recolhida a partir do vento através de turbinas projetadas para o efeito para o efeito. As turbinas, são normalmente colocadas em sítios ventosos, pois a produção de energia aumenta consoante o vento recebido até um certo limite de quilómetros por hora. A imagem seguinte ilustra uma turbina eólica.



Figura 38-Torre eólica

O funcionamento da turbina é o oposto do funcionamento de uma ventoinha. O vento, faz girar as pás que por sua vez giram um veio que está ligado a um gerador e produz energia. A imagem seguinte mostra o interior de uma turbina.

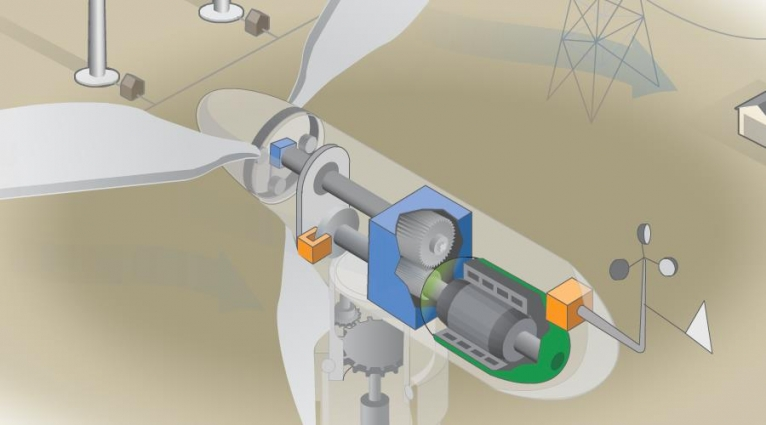


Figura 39-Interior de uma turbina

Podemos ver o veio ligado às pás que por meio de uma caixa desmultiplicadora (dentro da caixa azul) liga a um gerador.

Normalmente existem parques com várias turbinas em que todas utilizam um esquema semelhante de energia e centralizam essa energia numa central do parque que a injeta na rede de energia do país para ser vendida .